PROGRAMACIÓN AVANZADA

MEMORIA

Miguel Sande García Alejandro Sánchez Sánchez Brais Álvarez Sánchez

- 1. Arquitectura global.
- 2. Modelo.
 - 2.1. Clases persistentes.
 - $2.2.\$ Interfaces de los servicios ofrecidos por el modelo.
 - 2.3. Otros aspectos.
- 3. Interfaz gráfica.
- 4. Trabajos tutelados.
- 5. Problemas conocidos.

1. Arquitectura global.

Todo el proyecto está dentro de *pojo-app*, en *src* se encuentra todo el código, aquí nos encontramos con 3 paquetes: *sql*, *test* y *main*.

En sql se encuentran los scripts de creación de tablas y creación de datos.

En test nos encontramos con las pruebas de integración de la capa modelo y las pruebas funcionales contra la interfaz web.

En main se encuentra el resto del código.

Dentro de main se encuentran los paquetes java y resources.

En resources están los ficheros de configuración de spring e hibernate, así como los tmls de la interfaz web.

En java hay dos paquetes, model y web.

En model se encuentra todo lo correspondiente a la capa modelo (clases persistentes y sus servicios).

En web se encuentran las clases correspondientes a los tmls.

2. Modelo.

2.1. Clases persistentes.



 $\label{eq:Figure 1: Diagrama de clases.}$

2.2. Interfaces de los servicios ofrecidos por el modelo.

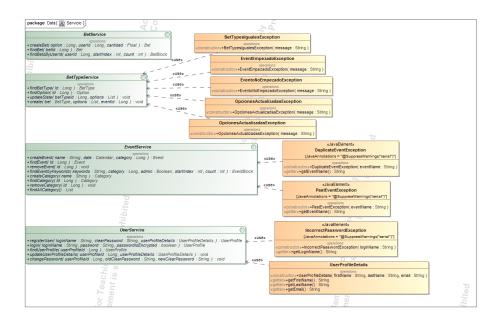


Figure 2: Servicios.

Se han agrupado los casos de uso de, por una parte betType y options, y por otra event y category. Esto es debido a que option y betType se usan de manera secuencial, por tanto lo mas lógico y eficiente es tenerlos juntos en el mismo servicio.

Y category y event porque solo la segunda utiliza a la primera.

2.3. Otros aspectos.

Se utilizan las siguientes excepciones de tipo checked:

- BetTypesIguales: evita duplicados.
- Unica Opcion
Ganadora: controla el número de opciones seleccionadas por el admin para que se correspondan con el atributo
 unica.
- OpcionesYaActualizadas: evita el renombrado de ganadoras.
- EventoEmpezado: evita apostar a eventos ya empezados.
- EventoNoEmpezado: evita poner cuales son las ganadoras antes del evento.
- EventoDuplicado: evita que se dupliquen los eventos.

Se modifica la case UserSession añadiendo el atributo admin para controlar si el usuario es administrador o un usuario normal, betType y options (una lista con las opciones), ya que una betType no puede existir sin opciones, de este modo podemos inicializarlo todo junto.

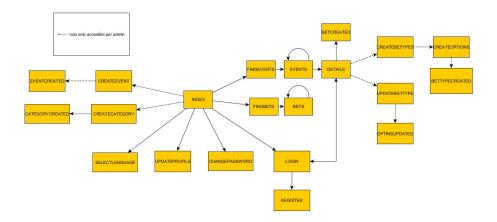


Figure 3: Navegación entre páginas.

Añadimos el patrón OpenSessionInView y el patrón Page-By-Page Iterator.

4. Trabajos tutelados.

Ajax.

Utilizamos Ajax para autocompletar los formularios de búsqueda, implementación de zonas en creación de betTypes y para mostrar los resultados de la búsqueda. Para formularios de búsqueda se añade la opción t:mixins=autocomplete en el Form, y en la clase java correspondiente se añade el método OnProvideCompletionsFromNameEvent.

Para la implementación de zonas se crean tres zonas, dos para formularios y una para mostrar la lista de opciones.

Para los resultados de búsqueda se crean dos zonas en el tml, una para el resultado y otra para los botones. La segunda zona es la responsable de actualizar la primera.

Pruebas funcionales.

Para realizar las pruebas se usa el componente WebDriver de Selenium. Para aislar estas pruebas de las pruebas de integración se crea un nuevo perfil maven. En el *pom.xml* se añaden las dependencias de Selenium y la configuración de cada perfil.

5. Problemas conocidos.

Al crear una betType duplicada no salta la excepción con Ajax.